PACKAGES IN JAVA

Packages are used in Java for organizing your class files. It can be considered as different folders to organize your files within your file system. You can put related class files together in one package. Java groups predefined classes into different packages like java.lang, java.util, java.io etc.

When no package is specified, the class is said to be in the default package, which is not a good programming practice.

To add your class to a package, you need to add a package structure definition to your class, compile to generate class file and then place your class file in a folder structure corresponding to your package definition.

You can write a package statement within your class using the package keyword:

package com.javajee.example;

A package statement should be the first java line in a java code (if present); only comments and whitespaces are allowed before that.

Java's built in packages start with java keyword (e.g. java.io.*, java.lang.*). User defined package are prohibitted to start with java.

You can start user defined package structure with the extension package javax, but it not preferred for normal application code.

Example Code: The example class used in this example is:

```
package com.javjee.example;
public class HelloWorld {

  public static void main(String[] args)

  {

    System.out.println("Hello World");

  }

}
```

Save this class in a file HelloWorld.java.

You can compile this using the javac command as:

```
javac HelloWorld.java
```

You should see HelloWorld.class inside if the compilation went well. After compiling, you need to execute the .class file. You need to place your class file in a folder structure corresponding to your package definition. So if your package definition is 'package com.javjee.example', then the folder structure would be com\javajee\example.

First, create (or decide on) a root folder to run the java command. We will create a folder called javaprograms. Inside that we will create folder structure as com\javajee\example and place the file HelloWorld.java in the example folder.

Now go to that folder (javaprograms) in command prompt and run the below command:

```
java com.javajee.example.HelloWorld
```

You should see the output as "Hello World".

Error 1: If you don't put the class file in the correct folder structure and try to run the above command, you will get an exception like below:

```
Error: Could not find or load main class com.javajee.example.HelloWorld
```

If you see this error, make sure you have put your HelloWorld.class under javaprograms\com\javajee\example and is running the statement from javaprograms folder.

Error 2: If you try to run without the qualifier (e.g. java HelloWorld instead of java com.javajee.example.HelloWorld), you will get an error like below:

```
Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorld (wrong name: com/javjee/example/HelloWorld)
        at java.lang.ClassLoader.defineClass1(Native Method)
    ...
```

If you see this error, make sure you have put the fully qualified name of the class when using java command.

Using - d option
Instead of compiling and then copying the .class file to the destination folder (com/javjee/example), you can ask javac command to create the folder structure and copy the .class file there using the -d option. For this, compile the source code as:

```
javac -d . HelloWorld.java
```

-d option should be follwed by a path. Dot (.) reprecent current directory.

If compilation is successful, this will create a folder structure as com/javjee/example under current directory (javaprograms) and the class file (HelloWorld.class) is copied to com/javajee/example. Now you can run the java command from the parent folder (javaprograms).

Using classpath

If you are executing from some other folder than the parent folder (javaprograms), then you can specify the parent folder in the classpath and run the java command.

```
java -classpath C:\javaprograms com.javajee.example.HelloWorld
```

You can give relative or absolute path.

More Examples

Consider that the class HelloWorld exists in the /apps/com/javajee/example directory. Assume the CLASSPATH environment variable is set to "." (current directory).

Valid executions:

$ java com.javajee.example.HelloWorld if run from the /apps directory

$ java -classpath /apps com.javajee.example.HelloWorld if run from any other directory

Invalid executions:

$ java HelloWorld if run from anywhere.

Remember the rule above that you need to specify the fully qualified name of the class from outside the package.

$ java com.javajee.example.HelloWorld if run from the /apps/com/javajee/example directory

Here we are executing from example directory and hence java will look for a directory corresponding to the qualified name as com/javajee/example/HelloWorld.class from the current directory (application directory).

To access a class or method in another package, you need to either use the fully qualified name of the file (e.g. com.javajee.example.HelloWorld) or you can use import statements.

Source : http://javajee.com/packages-in-java